Juliette Raubolt

ICS 661

Advanced AI

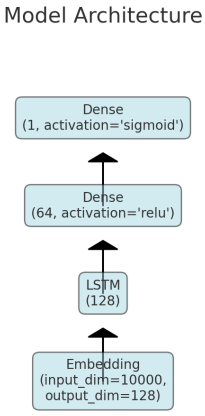
Fall 2024

Assignment 2 Report

Section 1: Task Description

Assignment 2 task is to classify the provided data (movie reviews) as positive or negative based on the text content. This involves text preprocessing, vocabulary extraction, and model training. Text preprocessing refers to cleaning the dataset. Vocabulary extraction refers to extracting a vocabulary set from the dataset which will be used as the input feature set for the model. Model training refers to training a Recurrent Neural Network (RNN) on the given data to classify reviews as positive or negative. The goal is to reach 75% accuracy.

Section 2: Model Description



Section 3: Experiment Settings

3.1 Dataset Description

The dataset used in this assignment includes 50,000 movie reviews split in half for training and testing. 25,000 are positive reviews and the other 25,000 are negative reviews. Negative reviews have a score of 4 or less out of 10 while positive reviews have a score of 7 or above out of 10. The organization of this data is split into two directories for training and testing and then within each of those directories there are two subdirectories for the positive and negative reviews.

3.2 Detailed Experimental Setups

For this assignment I first loaded the training, validation, and testing data. Since the different data was in separate files in subdirectories I used preprocessing.text-dataset_from_directory to load the data. Once the training, testing, and validation data was loaded I performed the necessary data preprocessing

including creating the vocabulary to be used (the most frequent 10,000 words) and vectorizing the data since we do not want the extra stop words and punctuation and we want all the text to be lowercase.

Once the data was ready I began creating the RNN model. I included an embedding layer to take in the data, one LSTM layer since this type of RNN works well with sequential data (like we have here). I then added an additional layer with 64 nodes that is fully connected and uses the relu activation function. Finally there is the output later with the sigmoid function since it works well for binary classification.

I then compiled the model with a binary cross entropy loss function, adam optimizer, and accuracy metric (precision, recall, and f1 score calculated since would run into issues when done here). At this point I trained the model, conducted the evaluation, and plotted the results.

3.3 Evaluation Metrics

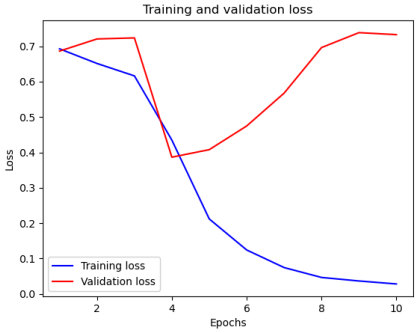
In the experiment, we will use Accuracy (how often the model was correct), Precision (how often positive is predicted correctly), Recall (true positives, how well the model can predict all the positives), F1 score (evaluation metric that combines precision and recall) as our evaluation metrics.

3.4 Source Code

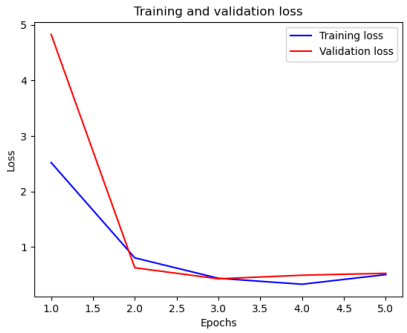
https://github.com/julietteraubolt/RNN-model/tree/main

3.5 Training Convergence Plot

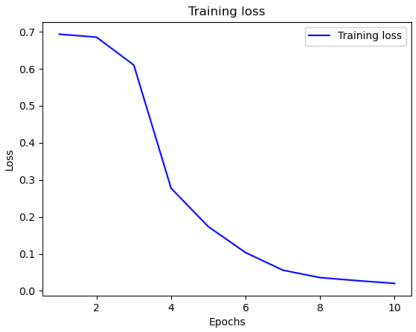
Unidirectional (baseline):



Bidirectional:



No Validation data:



3.6 Model Performance

Unidirectional (baseline):

- Test loss: 81.8%
- Test Accuracy:83.1%
- Precision: 49.8%
- Recall: 48.2%
- F1 Score: 49.0%

Bidirectional:

- Test loss: 56.1%
- Test Accuracy: 79.1%
- Precision: 49.6%
- Recall: 53.3%
- F1 Score: 51.4%

No Validation data:

- Test loss: 84.5%
- Test Accuracy: 84.2%
- Precision: 50.0%
- Recall: 48.8%
- F1 Score: 49.4%

3.7 Ablation Studies

Unidirectional (baseline):

- Text preprocessing with TextVectorization
- Embedding layer
- LSTM layer (unidirectional)
- Dense layer with ReLU activation and 64 nodes
- Output layer with sigmoid activation
- Binary crossentropy loss function
- Adam optimizer
- Accuracy, precision, recall, and F1 score metrics
- Epochs = 10

Bidirectional:

- Text preprocessing with TextVectorization
- Embedding layer

- Bidirectional LSTM layer
- Bidirectional LSTM layer
- Output layer with sigmoid activation
- Binary crossentropy loss function
- Adam optimizer
- Accuracy, precision, recall, and F1 score metrics
- Epochs = 5

No Validation data:

- No validation data (all training/testing data)
- Text preprocessing with TextVectorization
- Embedding layer
- LSTM layer (unidirectional)
- Dense layer with ReLU activation and 64 nodes
- Output layer with sigmoid activation
- Binary crossentropy loss function
- Adam optimizer
- Accuracy, precision, recall, and F1 score metrics
- Epochs = 10

I did three varying approaches to experiment with different features. The input and output layer for each are the same. For the baseline I used one unidirectional LSTM layer along with an additional layer with 64 nodes. For the next variation I used two bidirectional LSTM layers. For the third, the model was the same setup as the baseline but this time I used no validation data, but instead all data was used for training data to see the kind of impact this had. One thing to note is that I saw after the baseline model ran that 10 epochs were a lot so for the second model I just used five epochs. The overall best performing model was the third variation with the baseline setup and no validation data, perhaps the additional training data improved the model without running into the issue of overfitting. The worse performing was the second variation of the model with the two bidirectional LSTM layers. Further experimentation could be combining the two approaches to see what happens.